2022

Mining out of Delphi: bitcoin proofs of work without random oracles

BOSTON UNIVERSITY

GRADUATE SCHOOL OF ARTS AND SCIENCES

Dissertation

MINING OUT OF DELPHI:
BITCOIN PROOFS OF WORK WITHOUT
RANDOM ORACLES

by

MEGHNA SENGUPTA

B.Tech, Indian Institute of Technology Kharagpur, 2019
M.Tech, Indian Institute of Technology Kharagpur, 2019

Submitted in partial fulfillment of the

requirements for the degree of

Master of Science

2022

Approved by

First Reader

Leonid Reyzin, PhD
Professor of Computer Science

Second Reader

Ran Canetti, PhD
Professor of Computer Science

Acknowledgments

First, I would like to express my deep sense of gratitude and indebtedness to my supervisor, Prof. Leonid Reyzin for his guidance and support throughout the duration of my research work. He has given me so much of his time and taught me so much! I am also greatly thankful to Prof. Ran Canetti and Prof. Adam Smith for giving me their time and imparting their knowledge. I have learnt a lot from all three of my professors and for that I will always be grateful.

I am thankful to the entire family of the Computer Science department for providing all the facilities that were required for our work.

No good work is possible without friends and I have been extremely fortunate to have so many of them in my life. They have been incredibly supportive and were always there whenever I needed them. My heartiest thanks to all of them for always giving me joyous company.

I am extremely grateful to my partner in crime, Ken and to my Voovoo for bringing me so much happiness and joy over the past few years. Any crisis in my life, and there were a lot of those, could be dealt with just because of these two. They were always there with kind words and a smile to make my day.

Last but not the least, I thank my parents for their constant care and immense moral support. My mother is the biggest reason for me getting to where I am in life. Even though they were halfway across the world, I could always count on my parents for guidance on any and all matters whenever I needed them. Thank you for always having my back.

Meghna Sengupta

CS Department

MINING OUT OF DELPHI:

BITCOIN PROOFS OF WORK WITHOUT

RANDOM ORACLES

MEGHNA SENGUPTA

Boston University, Graduate School of Arts and Sciences, 2022

Major Professor: Leonid Reyzin, PhD
Professor of Computer Science

ABSTRACT

The Random Oracle (RO) model has proven to be very powerful for building cryptographic tools, but is known to be unrealistic. As such, finding security proofs for various crypto primitives that do not involve the random oracle model is of utmost importance. One property of the RO – Correlation Intractability (CI) – has proven to be an extremely useful property in this regard. A series of publications (Canetti et al., 2018) (Canetti et al., 2019) have constructed succinct publicly verifiable non-interactive argument systems using hash functions that are CI. Another application of RO is the Proof of Work(PoW) in Bitcoin where it serves many functions. In this thesis, we investigate these functions and attempt to eliminate the RO assumption for at least some of the functions that it serves.

For the puzzle itself in Bitcoin mining, we attempt to construct a candidate hash function that is moderately hard CI – a function for which, it is hard, but not impossible, to find, an input-output pair that satisfies a particular moderately sparse relation. The motivation for this is mainly the fact that such a function should be able capture the properties of the work that a successful bitcoin miner needs to do.

Contents

List of Abbreviations

CI	....	Correlation Intractability
FHE	....	Fully Homomorphic Encryption
RO	....	Random Oracle
SNARG	....	Succinct Non-interactive Argument
SNARK	....	Succinct Non-interactive Argument of Knowledge
ZK	....	Zero Knowledge

Chapter 1

Introduction

The twentieth century has seen the exponential rise of cryptocurrencies which has revolutionized the world of finance. Among them, Bitcoin has emerged as the leading decentralized cryptocurrency.

In the Bitcoin protocol, transactions are verified by nodes (which are part of the peer-to-peer network) using established cryptographic primitives along with a consensus algorithm, known as “proof of work”. The proof-of-work involves scanning for a value that when hashed, the hash, interpreted as an integer, is less than some (adjustable) value. (Nakamoto, 2008)

In existing literature, the hardness of the ”proof-of-work” of the Bitcoin protocol is proven with the assumption that the hash function used is in the Random Oracle model. However, this is controversial in the academic space since no known construction of a function is known that can satisfy all the properties that a random oracle is supposed to possess. (Canetti et al., 2004) This is a problem for all the practical applications that use random oracles in their proofs of security. Due to this, researchers have tried to distill the properties of the random oracle that are required for each application and come up with concrete constructions of hash functions that can be used in such applications and still be proven to be secure based on simple hardness assumptions. (Canetti et al., 2018) (Canetti et al., 2019)

We attempt to come up with an explicit function that can be used in the bitcoin protocol and prove that the probability of success in mining bitcoin is proportional to the computational power of a user.

1.1 Our Contributions

The problem of coming up with a security proof for the bitcoin backbone protocol in the standard model (without using a random oracle) can be reduced to finding a hash function that gives us the following assurances that we get from using the Random Oracle in Bitcoin -

1. A construction for a moderately hard puzzle — a puzzle that is going to be hard to solve in time less than a specific time T for all adversaries with a high probability. This is the puzzle that a user must solve in order to successfully add a block to the ledger.

2. Designing the puzzle in a way such that each puzzle gets uniquely determined by the contents of the block that the user is trying to add to the ledger.

3. A procedure to generate a large number of independent puzzles each with some fresh randomness. We must also be able to efficiently generate each fresh puzzle.

4. For all adversaries, generating a new puzzle from a new solution to an old puzzle (puzzle shopping) should be just as hard as solving a newly generated puzzle (puzzle solving).

5. A construction to enforce the sequentiality of the puzzles.

In Chapter 2, we discuss the construction of a hash function that we define as moderately correlation intractable. This function can be potentially used as the hash function in the puzzle for the bitcoin protocol. Finding a solution to the puzzle, in this scenario, would be equivalent to finding a correlation for the CI function wrt a specific relation. We also discuss what solving puzzles would entail, if we used this function to design them.

In Chapter 3, we attempt to tackle properties 3 and 5 from the list above. For property 5, we present two different ways to enforce the sequentiality of the large number of puzzles. However, both of these ways require random oracles for the security proof to hold. The proof mainly deals with the following two problems –

- *Generating many puzzles:* We present one way to generate a large number of puzzles by adding another parameter to the hash function key. This gives us a large number of different hash functions each of which can be used to create a fresh puzzle. Solving a specific puzzle would entail finding a correlation in the corresponding hash function.

- *Introducing randomness to each puzzle:* Currently, we use a Random Oracle to generate each puzzle which makes each fresh puzzle truly random.

For property 4, we do not know how to enforce the puzzle shopping constraint yet. The present proof, in Chapter 3, only works for puzzles that have unique solutions.

Chapter 2

Moderately Hard Correlation Intractability

As a candidate for a moderately hard Correlation Intractable function we use a special case of the construction used in (Canetti et al., 2018). The corresponding proof of CI till a certain point of time t is also very similar to the proof in (Canetti et al., 2018).

2.1 Construction of Hash Function

We consider the simple encryption scheme of One Time Pad with a PRG applied to the key. That is,

$$Enc(k, m) = m \oplus Exp(k) \quad \text{and} \quad Dec(k, c) = c \oplus Exp(k)$$

where $Exp() : 2^\lambda \rightarrow 2^n$ is a PRG expansion function.

R is considered to be a relation such that $(a, b) \in R$ iff b has k 0's as its last k bits.

Our hash function, $\mathcal{H}$, is then defined as follows -

$$\mathcal{H}_c(x) = Dec(x, c) = c \oplus Exp(x)$$

where $c \in \{0, 1\}^n$ and $x \in \{0, 1\}^\lambda$.

2.2 Moderate Correlation Intractability

In this section, we provide a proof that the function H_c is moderately Correlation Intractable. In order to simplify the proof, we do the analysis for the case where R is the relation where b has 0's in the last k bit positions. However, instead of the last k bit positions, we may also choose a different set of k bit positions which would give us a new relation R'. H_c is also correlation intractable wrt any such R'. The proof in that case will be exactly the same as the following proof for relation R.

Assumption 1: We assume that any adversary $\mathcal{A}$ that runs in time $t < 2^k$ and is provided with the last k bits of a PRG output, finds the corresponding seed with a probability of at most ϵ where $\epsilon \leq (poly(\lambda) \cdot 2^{\lambda-k})^{-1}$. This probability can be written as -

$$\Pr_{\substack{\alpha^* \\ c \leftarrow \text{last } k \text{ bits of } Exp(\alpha^*)}} [\mathcal{A}(c) \text{ outputs } \alpha^*] \leq \epsilon$$

Providing $\mathcal{A}$ with the last k bits of $Exp(\alpha^*)$ is equivalent to choosing a β^* that ends in k 0's uniformly and randomly and providing $\mathcal{A}$ with the XOR of β^* and $Exp(\alpha^*)$. Thus, the previous probability statement is equivalent to -

$$\Pr_{\substack{\alpha^* \\ \beta^* \text{ ends with } k\ 0's \\ c \leftarrow Exp(\alpha^*) \oplus \beta^*}} [\mathcal{A}(c) \text{ outputs } \alpha^*] \leq \epsilon$$

Theorem 1. *Let Assumption 1 be true. That is, the probability of any adversary that runs in time $t < 2^k$, finding the seed, given the last k bits of the PRG output is at most ϵ with $\epsilon \leq (poly(\lambda) \cdot 2^{\lambda-k})^{-1}$. Then $\mathcal{H}_c$ is correlation intractable till time $T = 2^k$ with respect to the relation R where λ is the length of the input .*

The theorem claims that till time $t < 2^k$, any adversary succeeds in finding a correlation for H_c wrt relation R with negligible probability.

Proof. Suppose towards a contradiction that there exists a probabilistic polynomial-time adversary $\mathcal{A}$ that breaks the correlation intractability of $\mathcal{H}_c$ at time $t < 2^k$ with a non-negligible probability ν. That is,

$$\Pr_{c}[(\mathcal{A}(c), c \oplus Exp(\mathcal{A}(c))) \in R] \geq \nu \tag{1}$$

The proof aims to show that if (1) is true, then the following inequality also holds -

$$\Pr_{\substack{\alpha^* \\ \beta^* \text{ ends with } k0's \\ c \leftarrow Exp(\alpha^*) \oplus \beta^*}} [\mathcal{A}(c) \text{ outputs } \alpha^*] \geq \frac{2^k \cdot \nu}{2^\lambda}$$

Since relation R is defined to be such that $(a, b) \in R$ iff b ends in k 0's, we can change the sampling of α^* and β^* to be uniformly sampled from R. Further, in the next step, instead of conditioning the sampling (α^*, β^*), we instead write the equivalent statement with the conditional in the probability statement itself:

$$\begin{aligned}\Pr_{\substack{\alpha^* \\ \beta^* \text{ ends with } k\ 0's \\ c \leftarrow Exp(\alpha^*) \oplus \beta^*}} [\mathcal{A}(c) \text{ outputs } \alpha^*] &= \Pr_{\substack{(\alpha^*, \beta^*) \in R \\ c \text{ s.t } Exp(\alpha^*) \oplus c = \beta^*}} [\mathcal{A}(c) \text{ outputs } \alpha^*] \\ &= \Pr_{\substack{(\alpha^*, \beta^*) \\ c \text{ s.t } Exp(\alpha^*) \oplus c = \beta^*}} [\mathcal{A}(c) \text{ outputs } \alpha^* | (\alpha^*, \beta^*) \in R]\end{aligned} \tag{2}$$

The denominator of the conditional is the probability that a uniformly chosen (α^*, β^*) belongs to relation R (that is, β^* ends in k 0's). This is evidently $\frac{1}{2^k}$.

β^* can then be replaced in the statement with $c \oplus Exp(\alpha^*)$ and α^* with $\mathcal{A}(c)$.

$$\begin{aligned}&\Pr_{\substack{(\alpha^*, \beta^*) \\ c \text{ s.t } Exp(\alpha^*) \oplus c = \beta^*}} [\mathcal{A}(c) \text{ outputs } \alpha^* | (\alpha^*, \beta^*) \in R] \\ =&\frac{1}{2^{-k}} \Pr_{\substack{(\alpha^*, \beta^*) \\ c \text{ s.t } Exp(\alpha^*) \oplus c = \beta^*}} [\mathcal{A}(c) \text{ outputs } \alpha^* \wedge (\alpha^*, \beta^*) \in R] \\ =&\frac{1}{2^{-k}} \Pr_{\substack{\alpha^* \\ c}} [\mathcal{A}(c) \text{ outputs } \alpha^* \wedge (\alpha^*, c \oplus Exp(\alpha^*)) \in R] \\ =&\frac{1}{2^{-k}} \Pr_{\substack{\alpha^* \\ c}} [\mathcal{A}(c) \text{ outputs } \alpha^* \wedge (\mathcal{A}(c), c \oplus Exp(\mathcal{A}(c))) \in R]\end{aligned} \tag{3}$$

In our last step, we relate the probability of the adversary $\mathcal{A}$ outputting a specific α^* to the probability of $\mathcal{A}$ outputting any string that is a correlation.

$$\begin{aligned}
&\frac{1}{2^{-k}} \Pr_{\substack{\alpha^* \\ c}}[\mathcal{A}(c) \text{ outputs } \alpha^* \wedge (\mathcal{A}(c), c \oplus Exp(\mathcal{A}(c))) \in R] \\
=&\frac{2^{-\lambda}}{2^{-k}} \Pr_{c}[(\mathcal{A}(c), c \oplus Exp(\mathcal{A}(c))) \in R] \\
= \geq & \frac{2^k \cdot \nu}{2^\lambda} \qquad (4)
\end{aligned}$$

Putting (2), (3) and (4) together, we have -

$$\Pr_{\substack{\alpha^* \\ \beta^* \text{ ends with } k\ 0's \\ c \leftarrow Exp(\alpha^*) \oplus \beta^*}}[\mathcal{A}(c) \text{ outputs } \alpha^*] \geq \frac{2^k \cdot \nu}{2^\lambda} \qquad (5)$$

Thus, the adversary $\mathcal{A}$ breaks Assumption 1 with probability $\epsilon \geq (1/\text{negl}) \cdot 2^{k-\lambda}$ which is a contradiction. This means that H_c is Correlation Intractable up till Assumption 1 holds. □

2.2.1 The Plausibility of Assumption 1

For a given $Exp()$ function and specific k bits, there are 2^k possible k bit combinations. Let us denote these combinations by $X_1, X_2, ..., X_{2^k}$. For each specific k-bit combination, X_i, let the size of the set of seeds whose PRG outputs match the k-bit combination be S_i. We are thus, effectively partitioning the set of 2^λ seeds into 2^k partitions. That is, $\sum_{i=1}^{n} S_i = 2^\lambda$.

For the following analysis, let us assume that each of the above defined parts are equal in size. That is, for any $i \in [1, n]$, $S_i = \frac{2^\lambda}{2^k} = 2^{\lambda-k}$.

Now, the success probability of an adversary for guessing a specific seed, given k bits of its PRG output, is going to be dependent on our PRG function, $Exp()$.

Best Case Scenario for adversary: The $Exp()$ function is really simple and it is possible for the adversary to sample a random seed that matches the given k

bits of output. This means that the adversary can correctly identify the part that the correct output belongs to. The best strategy for the adversary in this case is to randomly sample a seed from the part that it has correctly identified. Since in our assumption, the adversary gets a single chance to guess the seed, time t does not really help the adversary.

In this case, the probability that the adversary correctly guesses the specific seed will be equal to $\frac{1}{2^{\lambda-k}} = 2^{k-\lambda}$.

Thus, if the PRG function is really bad in terms of predictability, Assumption 1 is not plausible and the proof discussed in the previous section does not hold.

Worst Case Scenario for adversary: If the PRG function $Exp()$ is a Random Oracle then intuitively, it will be hard for any adversary to guess the specific seed from k bits of the output.

In this case, however, having time t does help the adversary. The adversary can make t queries to the random oracle and keep checking the query outputs to find one that matches the given k bits. It can then output the first query (if any) that matches the bits (is in the correct part). The best strategy that the adversary can adopt is to choose these t queries uniformly at random. Thus,

$$\begin{aligned}
\Pr[\text{None of the queries match the } k \text{ bits}] &= (1 - \frac{1}{2^k})^t \\
&= e^{-\frac{t}{2^k}} \\
&\approx 1 - \frac{t}{2^k}
\end{aligned}$$

$$\begin{aligned}
\Pr[\text{At least one of the } t \text{ queries matches the given } k \text{ bits}] &\approx 1 - (1 - \frac{t}{2^k}) \\
&= \frac{t}{2^k}
\end{aligned}$$

$$\begin{aligned}\Pr[\text{The query that matches the given } k \text{ bits is the specific seed}] &\approx \frac{\frac{t}{2^k}}{2^{\lambda-k}} \\ &\approx 2^{k-\lambda} \cdot \frac{t}{2^k} \\ &\approx \frac{t}{2^\lambda}\end{aligned}$$

So, as t tends to 2^k, the worst case probability of the adversary tends to the best case probability for the adversary. So, after time t, even the best possible PRG function will not render Assumption 1 plausible. Thus, our analysis for correlation intractability only holds till time at most $t < 2^k$.

Chapter 3

Sequentiality of puzzles

In this chapter, we try address some of the issues that need to be solved in order to use a moderately hard CI function as a puzzle in the context of Bitcoin.

3.1 Generating a large number of Puzzles

To be able to use our CI function as a puzzle, we need a way to generate a large number of fresh puzzles. For this, we introduce another parameter to the hash function key -

Let $I = \{p_1, ..., p_k\}$ be a set of k bit positions. For any a, let $\{a\}_I$ be the k bits that are present in a at the k bit positions $p_1, ..., p_k$.

We change the definition of the relation R to be such that $(a, b) \in R$ iff b is 0.

Our hash function, $\mathcal{H}$, is then defined as follows -

$$\mathcal{H}_{c,I}(x) = \{Dec(x, c)\}_I = c \oplus Exp(x)_I$$

where $c \in \{0, 1\}^n$, $x \in \{0, 1\}^\lambda$ and $I = \{p_1, ..., p_k\}$ with $p_i \in \{0, 1\}$.

This solves the problem of getting a large number of puzzles. However, the bigger problem is ensuring that a miner/user cannot solve a generated puzzle more easily once they have solved a certain bunch of other puzzles. We attempt to ensure this in the following sections. However, both of our approaches need the Random Oracle to work.

3.2 Sequentiality using RO and Encryption with universal-keys property

We consider a sequence of puzzles P_0, ..., P_n which have the solutions S_0, ..., S_n respectively. Additionally, we define a series of ciphertexts C_1, ..., C_n with

$$C_i = Enc_{RO(S_{i-1})}(P_i)$$

where the encryption scheme $\mathcal{E} = (Enc(), Dec())$ has the following properties -

- One-time CPA: $\mathcal{E}$ should possess indistinguishability under a one time chosen plaintext attack challenge (needed for claim 2).
- Universal-keys: For any given ciphertext and plaintext pair, (C, P), there exists a key k such that $Dec(k, C) = P$. Further, let $\mathcal{C}$ denote the distribution on the ciphertexts obtained by $Enc(k, P)$ for a uniformly chosen k and P. Let $\mathcal{K}$ denote the uniform distribution on the key space. Then if we pick C from $\mathcal{C}$, and a uniform P, there exists a way to sample k such that the distribution (C, k) is identical to $\mathcal{C} \times \mathcal{K}$ and $Dec(k, C) = P$. (needed for the proof of theorem 1)

Challenge: As inputs of the challenge, the adversary is provided with the first puzzle P_0 and all the ciphertexts C_1, ..., C_n. In order to solve the challenge, the adversary needs to output all the solutions S_0 to S_n.

Theorem 2. *If any given adversary solves a randomly generated puzzle in time $< t$ with probability at most α, then any adversary wins a challenge of n puzzles in time $< n(t - t')$, with probability at most $\frac{\alpha}{n} - \epsilon - \frac{q}{\delta}$.*

Here, ϵ is the semantic security advantage of the encryption scheme $\mathcal{E}$, q is the total number of queries that the adversary is allowed to make and δ is the size of the puzzle space.

t' is approximately the time needed to produce n encryptions and verify q puzzle solutions.

Proof:

We define a modified version of the adversary $\mathcal{A}$ that we call $\mathcal{A}'$. $\mathcal{A}'$ calls $\mathcal{A}$ and then at the end queries the Random Oracle with all the solutions that $\mathcal{A}$ outputs in order. It then outputs the same solutions.

We also assume that the random oracle calls are of the form (C_i, S_{i-1}). That is, the RO call specifies the ciphertext that the output is meant to be the key for, along with the previous solution.

Claim 1: $\mathcal{A}'$ makes at least n random oracle calls, specifically for the inputs S_0 to S_{n-1}.

Proof: Claim 1 is proved trivially since the modified algorithm makes n random oracle calls at the end in any case.

Claim 2: For all $0 \leq i \leq n-1$, let t_i be the time instance at which the adversary queries the Random Oracle on the solution S_i for the very first time. If the semantic security advantage of the encryption scheme $\mathcal{E}$ is ϵ for adversaries running in time $< t - t'$, then for $\mathcal{A}'$, the probability that i, j exist such that $j < i$ and $t_j > t_i$ is at most $n(\epsilon + \frac{q}{2^l})$.

Proof: Let us assume that the adversary does make the first RO call for S_i (at time t_i) before he makes the first random oracle call for S_j (at time t_j), that is, $t_i < t_j$. We claim that then, there exists a k such that the first RO call for S_{k+1} is made before the first RO call for S_k, that is, $t_{k+1} < t_k$.

Let t_{i-1} be the time stamp at which the adversary makes the first RO call for S_{i-1}. It may be that $t_{i-1} > t_i$, in which case, we have $k = i - 1$. Alternatively, $t_{i-1} < t_i$ which implies $t_{i-1} < t_j$. We can keep repeating this same argument until we have $t_{j+1} < t_j$ or we find a k such that $t_{k+1} < t_k$.

As proof of claim 2, we show a reduction of the adversary $\mathcal{A}'$ to an adversary $\mathcal{B}$ that breaks the semantic security of the encryption scheme $\mathcal{E} = (Enc(), Dec())$.

$\mathcal{B}$ finds the k at which $\mathcal{A}$ has the highest probability of having $t_{k+1} < t_k$. $\mathcal{B}$ then chooses $n+1$ puzzles P_0 to P_n and $n-1$ ciphertexts C_0, ..., C_k, C_{k+2}, ..., C_n. This determines the $n-1$ keys corresponding to each of the ciphertexts. Only the $k+1^{th}$ key is not known to $\mathcal{B}$. This is acceptable since $\mathcal{A}$ does not query the RO at S_k before S_{k+1} with probability ϵ. $\mathcal{B}$ then functions as follows -

1. $\mathcal{B}$ chooses P_B randomly and sends $P_A = P_{k+1}$ and P_B to the challenger and receives a challenge ciphertext C.

2. It then sets $C_{k+1} = C$ and sends P_0, C_1, ..., C_n to $\mathcal{A}'$.

3. On receiving a RO query (C_i, S) from $\mathcal{A}'$ where S is some solution, $\mathcal{B}$ verifies whether S solves problem P_{i-1}. If it does then $\mathcal{B}$ returns the i^{th} key.

4. $\mathcal{B}$ directly goes to step 5 when it receives the RO query (C', S') with $S' = S_k$. $\mathcal{B}$ verifies this by checking whether S' solves problem P_k. If no such query is received, $\mathcal{B}$ waits until $\mathcal{A}'$ stops running and then proceeds to step 5.

5. $\mathcal{B}$ then checks if any of the queries received by it solves either problem P_A or P_B. If a solution to P_B was queried at any point, $\mathcal{B}$ returns bit 1 (denoting P_B was encrypted). Otherwise, $\mathcal{B}$ returns 0 (denoting P_A was encrypted).

Let q be the total number of queries that $\mathcal{A}'$ is allowed to make.

And let the size of the total puzzle space be 2^l. This implies that Pr[A given solution is correct for a random puzzle] $= \frac{1}{2^l}$.

When P_A is encrypted, the view of $\mathcal{A}'$ is independent of P_B, and therefore chances that any one of the q solutions $\mathcal{A}'$ queries is a solution for P_B is $\frac{q}{2^l}$.

When P_B is encrypted, $\mathcal{B}$ outputs 1 whenever $\mathcal{A}'$ queries S_{k+1} before S_k. Since k was chosen at random, this probability is at most $\frac{\beta}{n}$.

Thus, given that the semantic security advantage of the encryption scheme is ϵ, we have,

$$
\begin{aligned}
\epsilon &= |\Pr[\mathcal{B} \text{ outputs 1 and } P_B \text{ was encrypted}] - \Pr[\mathcal{B} \text{ outputs 1 and } P_A \text{ was encrypted}]| \\
&\geq \Bigg| \frac{\Pr[\mathcal{A}' \text{ makes some RO call out of order }]}{n} \\
&\qquad - q \times \ \Pr[\text{ A given solution is correct for a random puzzle }] \Bigg| \\
&\geq \left| \frac{\Pr[\mathcal{A}' \text{ makes some RO call out of order }]}{n} - \frac{q}{2^l} \right| \qquad (6)
\end{aligned}
$$

Assuming that the probability that $\mathcal{A}'$ makes a RO call out of order is $> \frac{nq}{2^l}$, the following inequality can be obtained from statement (6):

$\Pr[\mathcal{A}'$ does some RO call out of order$] \leq n \times (\epsilon + \frac{q}{2^l})$

This proves Claim 2.

Proof of Theorem 2: Let $\mathcal{A}$ be an adversary that wins a challenge of $n+1$ puzzles in time $< (n+1)t$ and makes all its random oracle queries in order with probability α.

We can then build an adversary $\mathcal{B}$ that solves a randomly generated puzzle given to it in $< t$ time. This breaks our assumption and leads us to a contradiction.

Since adversary $\mathcal{A}$ takes $< (n+1)t$ time and makes the n RO calls in order, there must exist an interval marked by two consecutive random oracle calls in which $\mathcal{A}$ solves a puzzle and takes $< t$ time. The position of the puzzle that $\mathcal{A}$ is most likely to take $< t$ time to solve is computed along with the set of starting puzzles. This position is denoted by r and the puzzles by $P_0, ..., P_{r-1}$. The probability that $\mathcal{A}$ takes $< t$ time to solve the r^{th} puzzle must be $> \frac{\alpha}{n}$.

n random ciphertexts are generated and $\mathcal{A}$ is run with some $n+1$ puzzles of which P_0 to P_{r-1} are the first $r-1$ puzzles. The state of $\mathcal{A}$ after it solves the first $r-1$ puzzles is saved and hardwired into $\mathcal{B}$. Any RO queries from $\mathcal{A}$ during this run are answered with the corresponding keys.

Note: The previous step makes the reduction non-uniform. We may also model the adversary $\mathcal{B}$ as having an offline pre-computation phase where it does the above by itself.

Following this construction, $\mathcal{B}$ then interacts with its challenger as follows -

1. $\mathcal{B}$ is given an independent random puzzle by the challenger. Let this puzzle be P.

2. $\mathcal{B}$ programs the Random Oracle to output a key k_r as the output to the solution S_{r-1} such that the r^{th} puzzle is P. It then runs its saved state of $\mathcal{A}$ with P as the r^{th} puzzle.

3. $\mathcal{B}$ wins if $\mathcal{A}$ returns the correct solution S in time $< t$.

Pr[$\mathcal{B}$ solves given puzzle in $< t$]

$\geq \frac{1}{n}$ Pr[$\mathcal{A}$ solves $n+1$ puzzles in $< (n+1)t$ and makes all RO calls in order]

$\geq \frac{1}{n}$ [Pr[$\mathcal{A}$ solves $n+1$ puzzles in $< (n+1)t$] - Pr[$\mathcal{A}$ solves $n+1$ puzzles in $< (n+1)t$ and makes at least one out of order RO call]]

$\geq \frac{1}{n}[\alpha - n(\epsilon + \frac{q}{2^l})]$

$\geq \frac{\alpha}{n} - \epsilon - \frac{q}{2^l}$

Note: The step 2 of the functionality of the adversary $\mathcal{B}$, requires the encryption scheme to have the universal-keys property. For any randomly chosen ciphertext, there must exist a key that decrypts the ciphertext to the given random puzzle. This property essentially gives a One-Time-Pad like encryption scheme. In section 3.2.1, we explore this property further.

The following two points are to be noted regarding this -

1. An FHE scheme breaks this reduction. All the ciphertexts can be combined to make a single ciphertext using the FHE and the computation required by the puzzle in that case may be much less costlier than doing n individual smaller computations.

2. This implies that any encryption scheme having universal-keys property cannot be FHE.

3.2.1 Proof of Claim 2 with Perfect Security

The second property of the encryption scheme — universal-keys — is needed for the main proof of the theorem to work.

Claim 1. *If the simulation* (P, C_r) *pairs by* $\mathcal{B}$ *is perfect, then the encryption scheme must be perfectly secure.*

Proof. The view of the adversary $\mathcal{A}'$ — which is a distribution on ciphertexts, keys, and puzzles — the must be the same in the reduction simulation as it is in real-life. In real life, keys and puzzles are chosen uniformly, and ciphertexts are computed from them. The simulation is the same, except for the r-th ciphertext: C_r is chosen first, then a puzzle P is chosen uniformly, and then $\mathcal{B}$ outputs k_r such that $Enc(k_r, P) = C_r$. Thus, in the simulation P is independent of C_r, and so it must also be in real life, where P is chosen uniformly. Thus, the encryption scheme needs to have the property that for a uniform P, given a ciphertext of P, P remains uniform. This property is equivalent to perfect secrecy [as proved in Theorem 2.4 of (Boneh and Shoup, 2015)]. $\square$

This implies that the encryption scheme $\mathcal{E}$ will need to be perfectly secure which is a much stronger property than the one-time CPA security which we needed to prove claim 2. Thus, there is a simpler alternative proof for claim 2 which utilises the fact that $\mathcal{E}$ needs to be perfectly secure.

The statement of the main theorem remains almost the same in this case as in Theorem 2 in Section 3.2, with $\epsilon = 0$ in the probability value. That is, any adversary wins a challenge of n puzzles in time $< nt$ with probability $\frac{\alpha}{n} - \frac{q}{2^l}$.

3.3 Sequentiality using only RO

We now consider a sequence of puzzles P_0, ..., P_n which have the solutions S_0, ..., S_n respectively. Additionally, we have the condition that

$$P_i = RO(S_{i-1})$$

We, again, define a modified version of the adversary $\mathcal{A}$ that we call $\mathcal{A}'$. $\mathcal{A}'$ calls $\mathcal{A}$ and then at the end queries the Random Oracle with all the solutions that $\mathcal{A}$ outputs in order. It then outputs the same solutions.

Similar to the previous section, **Claim 1** is therefore trivially true since $\mathcal{A}'$ makes n RO calls at the end.

Claim 2: Let us assume that $\mathcal{A}'$ makes the call $RO(S_{k+1})$ before making the call $RO(S_k)$.

The adversary then makes the call for S_{k+1} without having any knowledge of the problem P_{k+1}. The adversary's best strategy in that case is to make queries for the most probable solutions in the solution space.

Thus, in this case,

Pr[$\mathcal{A}'$ does some RO call out of order] $\leq \frac{q}{2^l}$

The proof of the main theorem proceeds with the exact same arguments as before, except in this case, the probability analysis yields the result that -

Pr[$\mathcal{B}$ solves given puzzle in time $< t$] $\geq \frac{\alpha}{n} - \frac{q}{n \cdot 2^l}$

3.4 Sequentiality for Expected Time of solving puzzles

So, in the bitcoin setting, instead of a lower bound on puzzle-solving time, we have a guarantee only on expected puzzle solving time, because the probability of finding a solution grows linearly with time. Let's say $\mathcal{A}$ solves a sequence of n puzzles faster than expected time nt, we can then reduce it to the one puzzle adversary $\mathcal{B}$ solving one puzzle in time faster than expected time t. This does not give us a contradiction however.

We can instead analyze the probability of $\mathcal{A}$ solving n puzzles faster than time nt, and reduce that to the probability of $\mathcal{B}$ solving a single puzzle in time lesser than t.

Claims 1 and 2 from our analysis of worst-case time (in sections 4 and 5) still hold true in this current analysis. We thus have the result that any given adversary $\mathcal{A}$ makes at least n random oracle calls which are all in order with a probability of $1 - n(\epsilon + \frac{q}{2^l})$

For the main theorem, we restate it to the following for the expected time version of the problem -

Theorem 3. *Let the probability that any adversary takes time $< t'$ to solve a random puzzle be α. Then any adversary $\mathcal{A}$ has a probability of winning a challenge of n puzzles in time $< nt'$ of at most $\frac{\alpha}{n} - \epsilon - \frac{q}{2^l}$.*

3.5 The problem with replacing of Random Oracles

3.5.1 Using a Correlation Intractable hash function to generate puzzles

Initial approach: The first idea that we had to generate new puzzles as well as tie each new puzzle to previously solved puzzles was by chaining the puzzles together. For this chaining, we thought of applying a Correlation Intractable (CI) hash function to the solution of a current puzzle that would output the next puzzle in the chain.

The security proof in this case would then require each of the puzzles in the chain to be equally hard. In other words, solving one puzzle should not help an adversary find a next puzzle that is easy to solve. That is, the CI function should not generate a puzzle that is easy to solve for some adversary. Using an inductive logic, the CI should also avoid outputting a puzzle that would make any puzzle down the chain easy to solve.

However, Correlation Intractability is defined such that it is hard for all adversaries to find correlations for a randomly chosen hash function that is independent of the relation. In our case, the hash function and the relation are not independent of each other, which is why CI cannot be used for chaining.

Problem of getting randomness: The puzzles obtained from the CI hash function are all determinable in advance by the adversary. This is at odds with the requirement that each new puzzle must have some fresh randomness in it. Otherwise, the adversary, when they are given the first puzzle, can find multiple solutions in order to look for a solution that will in turn provide them with an easier new puzzle. And since the generation of a new puzzle does not require a query to an oracle, theoretically, it becomes extremely difficult to prove anything about the time taken to solve a puzzle.

3.5.2 Using a punctured PRF in place of RO

When using a punctured PRF, we no longer have the advantage of knowing exactly when the adversary finds a solution to a puzzle exactly, since no queries are made to compute the next puzzle. This brings us to the more basic problem of replacing a Random Oracle. Since time stamps can no longer be defined by oracle queries in the absence of a random oracle, it is difficult to define the exact amount of time taken by a user to solve any puzzle in a sequence of puzzles.

References

Boneh, D. and Shoup, V. (2015). *A Graduate Course in Applied Cryptography*.

Canetti, R., Chen, Y., Holmgren, J., Lombardi, A., Rothblum, G. N., Rothblum, R. D., and Wichs, D. (2019). Fiat-shamir: From practice to theory. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2019, page 1082–1090, New York, NY, USA. Association for Computing Machinery.

Canetti, R., Chen, Y., Reyzin, L., and Rothblum, R. (2018). *Fiat-Shamir and Correlation Intractability from Strong KDM-Secure Encryption*, pages 91–122.

Canetti, R., Goldreich, O., and Halevi, S. (2004). The random oracle methodology, revisited. *J. ACM*, 51(4):557–594.

Nakamoto, S. (2008). Bitcoin: A peer-to-peer electronic cash system.

CURRICULUM VITAE

[redacted]